What is claimed is:

1. A method to accommodate two different data structures when porting a protocol stack to a driver, comprises:

providing entries in a driver packet buffer structure to mimic a buffer structure of a ported protocol stack; and

providing entries in the buffer structure of the ported protocol stack.

2. The method of claim 1 wherein providing entries in the driver packet buffer structure comprises adding a flag entry to a data block of the driver packet buffer structure.

3. The method of claim 2 wherein the flag entry identifies any buffer generated in the driver and outside of the protocol stack.

4. The method of claim 1 wherein providing entries in the driver packet buffer structure comprises adding a pointer-to-header entry to a data block of the driver packet buffer structure.

5. The method of claim 4 wherein the pointer-to-header entry determines an appropriate freeing routine.

6. The method of claim 1 wherein providing entries in the buffer structure of the ported protocol stack comprises appending a flag entry to a message block of the buffer structure of the ported protocol stack.

7. The method of claim 1 wherein providing entries in the buffer structure of the ported protocol stack comprises appending a pointer-to-header entry to a data block of the buffer structure of the ported protocol stack.

8. The method of claim 1, wherein providing entries in the driver packet data structure comprises:

appending a data block of the driver packet data structure to have the same pointers as in a message block of the buffer structure of the ported protocol stack.

9. The method of claim 1, wherein providing entries in the driver packet data structure comprises:

appending a data block of the driver packet data structure to have the same entries as in a data block of the buffer structure of the ported protocol stack.

10. The method of claim 1, wherein providing entries in the driver packet data structure comprises:

appending a data block of the driver packet data structure to have the same data as in an actual data buffer of the buffer structure of the ported protocol stack.

11. An apparatus comprising:

a memory that stores executable instructions for accommodating two different data structures when porting a protocol stack to a driver; and

a processor that executes the instructions to:

provide entries in a driver packet buffer structure to mimic a buffer structure of a ported protocol stack; and

provide entries in the buffer structure of the ported protocol stack.

12. The apparatus of claim 11 wherein to provide entries in the driver packet buffer structure comprises adding a flag entry to a data block of the driver packet buffer structure.

13. The apparatus of claim 12 wherein the flag entry identifies any buffer generated in the driver and outside of the protocol stack.

14. The apparatus of claim 11 wherein to provide entries in the driver packet buffer structure comprises adding a pointer-to-header entry to a data block of the driver packet buffer structure.

15. The apparatus of claim 14, wherein the pointer-to-header entry determines an appropriate freeing routine.

16. An article comprising a machine-readable medium that stores executable instructions for accommodating two different data structures when porting a protocol stack to a driver, the instructions causing a machine to:

provide entries in a driver packet buffer structure to mimic a buffer structure of a ported protocol stack; and

provide entries in the buffer structure of the ported protocol stack.

17. The article of claim 16 wherein to provide entries in the driver packet buffer structure comprises adding a flag entry to a data block of the driver packet buffer structure.

18. The article of claim 17 wherein the flag entry identifies any buffer generated in the driver and outside of the protocol stack.

19. The article of claim 16 wherein to provide entries in the driver packet buffer structure comprises adding a pointer-to-header entry to a data block of the driver packet buffer structure.

20. The article of claim 19 wherein the pointer-to-header entry determines an appropriate freeing routine.